Winter 12-1-1993

Autonomous Exploration and Control of Chaotic Systems ; CU-CS-691-93

Elizabeth Bradley
University of Colorado Boulder

AUTONOMOUS EXPLORATION AND CONTROL OF CHAOTIC SYSTEMS

Elizabeth Bradley

CU-CS-691-93

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Autonomous Exploration and Control of Chaotic Systems

Elizabeth Bradley

Department of Computer Science
University of Colorado
Campus Box 430
Boulder Colorado 80309-0430

CU-CS-691-93 December 1993

In review for the *Communications of the ACM*.

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE AGENCIES NAMED IN THE ACKNOWLEDGEMENTS SECTION.

Autonomous Exploration and Control of Chaotic Systems

Elizabeth Bradley

lizb@cs.colorado.edu
Department of Computer Science
University of Colorado
Boulder CO 80309

Abstract — Control algorithms that exploit chaotic behavior can vastly improve the performance of many practical and useful systems. Phase-locked loops, for example, are normally designed using linearization. The approximations thus introduced lead to lock and capture range limits. Design techniques that are equipped to exploit the real nonlinear nature of the device loosen these limitations. The program *Perfect Moment* is built around a collection of such techniques. Given a differential equation and two points in the system's state space, it automatically selects and maps the region of interest, chooses a set of trajectory segments from the maps, uses them to construct a composite path between the points, and causes the system to follow that path via appropriate parameter changes at the segment junctions. Rules embodying theorems and definitions from nonlinear dynamics are used to limit complexity by focusing the mapping and search on the areas of interest. Even so, these processes are computationally intensive. However, the sensitivity of a chaotic system's state-space topology to the parameters of its equations and the sensitivity of the paths of its trajectories to the initial conditions make this approach rewarding in spite of its computational demands. Controlled trajectories found by this program exhibit a variety of interesting and useful properties. For example, detours through chaotic regions can be used to steer trajectories across boundaries of basins of attraction, effectively altering both the geometry of and convergence properties within a particular convergence region — such as the capture range of a phase-locked loop circuit.

Keywords and Content Indicators: artificial intelligence, problem-solving environments, nonlinear dynamics, controlling chaos, nonlinear control.

Introduction

This paper presents a control system design methodology that actively exploits chaotic behavior and that is carried out autonomously by a computer program. This tack not only broadens the field of nonlinear control to include the class of systems brought into vogue — and focus — by the last few decades of interest in chaos, but also opens a new angle on many old problems in the field. Many of these problems, new and old, are interesting and useful applications; all exhibit intricate and powerful behavior that can be harnessed by suitably intelligent computer programs.

The algorithms presented here intentionally route systems through chaotic regions, using extensive simulation, qualitative and quantitative reasoning about state-space features, and heuristics drawn from nonlinear dynamics theory to navigate through the state space. This approach is a sharp contrast to traditional control theory tactics, most of which avoid chaotic behavior at all costs. As these regions often comprise a large and rich part of a chaotic system's state space, avoiding them constrains a system to a possibly small and comparatively boring part of its range of operation.

The program that embodies these algorithms, *Perfect Moment*, constructs reference trajectories in advance, using a model of the target system. The controller thus designed is then used for on-line, real-time control of the system. During the generation of the reference trajectory, segments are selected from a collection of automatically constructed state-space portraits and spliced together into a path that meets the specified control objectives. An on-line controller causes the system to follow this segmented path via judicious parameter value switches at the segment junctions. In the process of constructing and examining the state-space portraits, the mapping and search algorithms use domain knowledge — rules that capture theorems and definitions from nonlinear dynamics — to choose both the trajectory distribution on each portrait and the parameter spacing between portraits so that the collection is a representative sampling of the system's dynamics.

Nonlinearity can provide significant leverage to a control algorithm that is designed to find and exploit its inherent sensitivity to parameter and state variation. The outcome of such tactics resembles, in spirit, the paradigms of Maxwell's Demon and Simon's Ant, wherein environmentally available energy is exploited via small, well-chosen control actions. Small *errors* can, however, have equally dramatic effects. This leverage is the power of and, paradoxically, the difficulty with nonlinearity.

Chaos provides some additional advantages beyond simple nonlinear leverage. The density with which trajectories cover a chaotic attractor has obvious implications for reachability. Its structural stability in the presence of state noise endows a chaotic attractor with a measure of robustness that can be used to counterbalance the exponential error growth mentioned above. Furthermore, these attractors contain an infinite number of unstable periodic orbits that can be located and stabilized.

The goal of this research is to identify and characterize some of these useful properties, to work out some computer control algorithms that take advantage of them, and to demonstrate their effectiveness on some practical examples. The task is the classic control problem: to cause a system to travel from one specified state-space point to another in some *optimal* way, where optimal is defined by the user and the application. The domain of application is the set of dissipative chaotic systems that have a single control parameter, are observable, and operate under well-specified design constraints. The practical examples in this paper are drawn from mechanical and electrical engineering, but chaos appears in virtually every branch of applied science, so potential applications are by no means sparse.

Striking results have been achieved with these techniques[2, 4, 5]: a very small control action, delivered precisely at the right time and place, can accurately direct the system to a distant point on the state space — hence the program's name. An equally small change can be used to move from the basin of attraction of one distant fixed point to the basin of another. Control actions can briefly push a system directly away from the objective in order to reach a globally superior path to that point; "strange attractor bridges" can open conduits to previously unreachable regions.

The next section begins with a high-level description of how *Perfect Moment* works, then covers each stage of its analysis, synthesis and control phases in more detail. The capabilities and drawbacks of the program are illustrated with three examples — the Lorenz system, the driven pendulum, and the phase-locked loop — and the paper concludes with a discussion of caveats and possible extensions.

How It Works

Perfect Moment is presented with a nonlinear ordinary differential equation (ODE), some control objectives (an origin, a destination, a tolerance, and a specification of optimality cost), and a control parameter range. The program autonomously explores the system's behavior, manipulating the control parameter and the search region during its explorations, identifying and exploiting nonlinear and chaotic features and properties in the course of the process. Filtering the results of this exploration through the specified optimality cost, it builds a segmented *reference trajectory* between the origin and the destination. Finally, its real-time section executes the control actions that cause the target system to follow that trajectory.

Perfect Moment produces a running commentary on its status, actions and choices. This feature is more than a development aid. An experienced user can monitor this report and intervene to accelerate the process or to push on some particular part of the design. Just as importantly, a novice (and occasionally even the program's author) can learn from watching the program's actions: about nonlinear dynamics, about control, about AI tactics like searching, sorting and multiple-scales processing — and of course about the system in question.

All of the algorithms covered in this section are described and illustrated in much more detail in [3].

Exploring the System's Behavior

The task of *Perfect Moment*'s mapping module is to construct a set of portraits that is a representative sample of the system's dynamics. This entails the recognition of regions where parameter changes cause bifurcations and other interesting effects, as well as the selection of a set of trajectories that efficiently captures the dynamics at a particular parameter value. Both of these problems are relatively easy for a human expert and very difficult to mechanize. *Perfect Moment* solves both with a state-space grid, which partitions an n-dimensional state-space region into n-parallelepipeds via linear division of each state variable axis. This method has a long and rich history, both in dynamics[13] and in AI[16]. Discretization of trajectories on such a grid not only allows them to be represented with far less information than their floating-point counterparts, but also facilitates the type of rough-first, fine-later reasoning that is the basis of this and many other efficient AI problem-solving techniques.

The mapper first chooses the region of interest by expanding the bounding box of the origin and destination. This expansion allows the program to explore counterintuitive moves — path segments that send the system away from the apparently "correct" direction in order to reach a faster overall path. This state-space region is then divided into parallelepipeds, as described in the previous paragraph. Defaults for the overrange factor and the cell size, which govern the expansion and division of the region, are built into the program; these values were chosen after experiments on several systems, but may be specified by the user as well. Both of these parameters are the objects of extensive dynamics-driven manipulation in the later stages of the program — for instance, the cell size is lowered in highly turbulent regions and the region is expanded if the segment search fails. These later refinements allow *Perfect Moment* to recover from many initial bad choices.

Given this setup, a single state-space portrait consists of the set of trajectories that emanate from the centers of all cells in the grid, integrated with fourth-order Runge-Kutta[21] until they leave the region or relax to an attractor.[1] See figure 1 for an example. The origin and destination points are used as initial conditions in the cells that contain

[1]The integration length is limited by another heuristically chosen, dynamically adapted parameter; the time step is chosen via the results of an *adaptive* integrator run and then lowered as dictated by the dynamics, the cell size, and the search mechanics. See section 4.2.1 of [3] for more details.

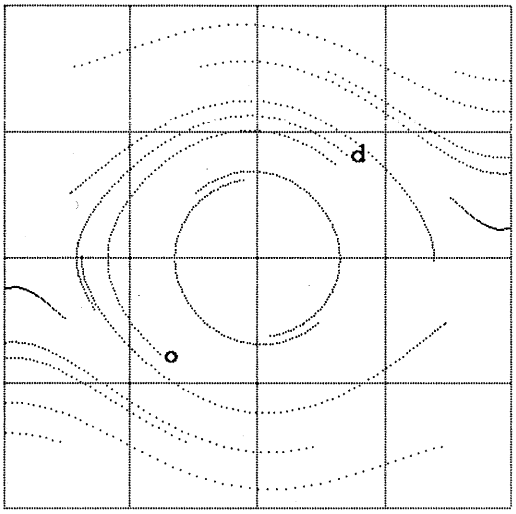

Figure 1: State-space portrait of the damped pendulum: the bounding box of the origin and destination is expanded to determine the region to be mapped, and one trajectory is generated from each cell center

them;[2] note the trajectories in the figure that start at the points marked "o" and "d." During the construction of an individual portrait, the role of the grid is to guide the selection of a *representative* set of trajectories from among the uncountably infinite number of candidates: the cells are chosen small enough so that each part of the region is explored, and yet large enough to restrict the amount of information to the minimum necessary to capture the dynamics.

Perfect Moment classifies the dynamics of a portrait in terms of the sets of cells to which its trajectories relax. Four types of attractors exist in dissipative chaotic systems: fixed points, limit cycles, quasiperiodic orbits, and chaotic attractors. Each has a characteristic signature in the discretized version of a trajectory:[3] for instance, a trajectory that has reached a fixed point exhibits a long terminal time span in a single cell, while a limit cycle appears as a repeating sequence of cells. The finite extent and discretization of the region under investigation, as well as the time step and trajectory length, alter the apparent properties of these attractors; for instance, a "fixed point" could really be a small chaotic attractor enclosed by a single cell, and a lightly damped fixed point might not be recognized in a short trajectory.

[2]the latter is integrated backwards in time

[3]A discretized trajectory is the sequence of grid cells touched by the floating-point trajectory, together with time of entry/exit.

The roughness of this dynamics classification scheme is intentional: it allows the program to adapt the analysis scale to the task at hand. Recall that these portraits are to be searched for path segments. If one is trying to select a section of interstate highway from San Francisco to Denver, effort devoted to recognition of county road junctions in rural Nevada is wasted. Of course, the rough grain can cause problems: for instance, an attractor whose basin encloses no cell centers may escape notice, thin-band chaotic attractors are sometimes misclassified as limit cycles, etc. Most of these problems can be solved by the additional processing that is driven by the refinement loops in the search phase of the program.

These grid-based classification results are coupled with standard binary search to space portraits out along the parameter axis in a pattern that samples all of the interesting and useful dynamics within the specified range. *Perfect Moment* first constructs portraits at a coarse parameter spacing, then recursively bisects the parameter range between any neighboring portraits that differ in attractor topology or in proximity of an attractor to the control objective.[4] Of course, any earlier classification errors can cause this algorithm to overlook important dynamics and construct an incomplete stack of portraits. The KAM program[24], which performs a similar analysis of the state-space features of Hamiltonian systems, is specifically designed to avoid this type of problem. It uses vision techniques, rather than cell patterns in discretized trajectories, to classify dynamics, and then processes those results with powerful nonlinear dynamics rules to infer when structures have been overlooked. MAPS[25] uses another interesting construct, the *flowpipe*, to classify trajectories in an equivalence class,[5] using a powerful nonlinear dynamics theorem to determine the boundary of the pipe and representing the pipe geometrically with Delaunay triangulation.

The output of the machinery described in this section is a set of portraits, like those in the schematic in figure 2, that is a representative sample of the system's behavior. The distribution and characteristics of the trajectories on each member of the set are chosen automatically so that each individual portrait is both representative of the dynamics at that parameter value and recognizable to the program. Multiple-scales dynamics classification and range bisection are used to zero in on bifurcations and other points of interest. Nonlinear dynamics knowledge guides the process to a solution that meets these requirements without excessive computational complexity.

Building a Reference Trajectory

[4]This zeroing-in process is also limited by a user-specified iteration depth, which can be used to bypass some or all of the dynamics classification or to let the user limit accuracy for, e.g., a first cut at a design.

[5]a flowpipe comprises all trajectories that are relaxing to a particular fixed point.

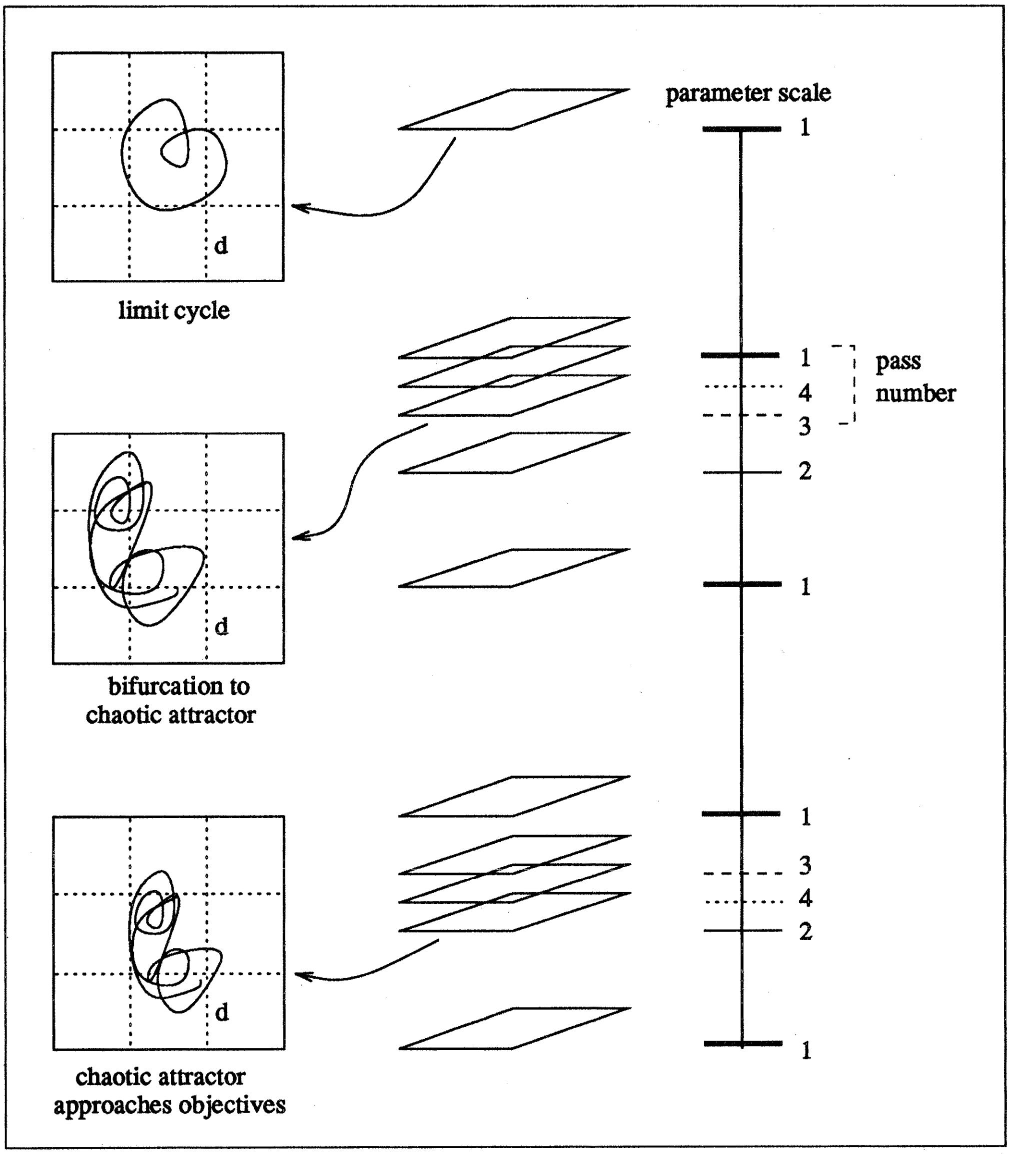

Figure 2: A stack of state-space portraits produced by the mapper: the initial parameter step was one-fourth the parameter range (the vertical line) and the iteration depth was 4. The mapper reduced the parameter step in two ranges, once because of a bifurcation and once because an attractor entered the destination (**d**) endpoint cell

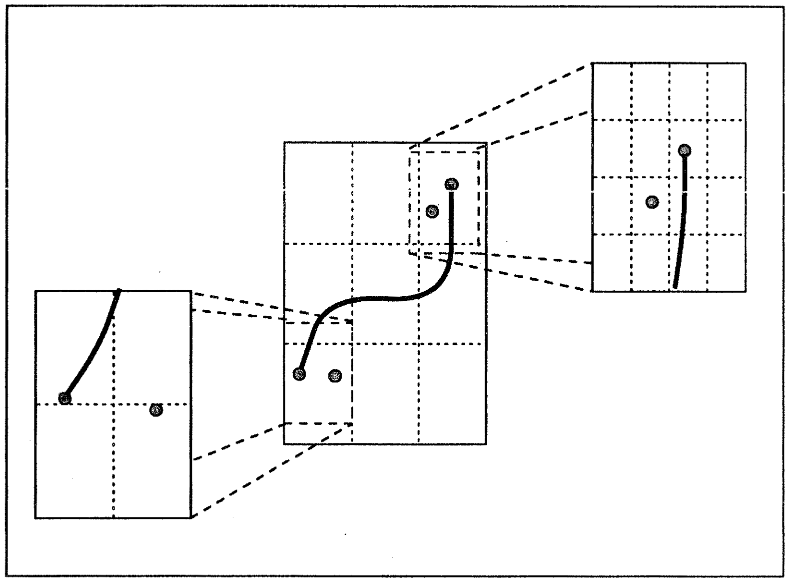

Figure 3: Refining regions between search passes: a gross path between origin and destination cells is first found, then each endpoint cell is recentered around the pair of points that will be an [origin destination] pair on the next search pass. A new grid division is then computed in each region, reflecting the local dynamics therein

The path finder searches a set of state-space portraits for trajectory segments that meet the control objectives. The planning algorithm that it uses to connect two points resembles that of the GPS[18]. A rough path is first found between the regions surrounding the origin and destination. The program then attempts to find paths that bridge the gaps between the endpoints of this *core segment* and the control objectives, recursively refining the reference trajectory until the control tolerance is met.[6] The grid plays a variety of roles in the path-finding process: it is used to define the regions around the origin and destination (the *endpoint cells*), to focus all processing at the appropriate scale, and to channel the control flow of the search.

An *optimality metric* assigns a weight to each cell: this metric is a Scheme procedure, and thus is easy to program and to change. All of the examples in this paper minimize time and path length; in other applications, one might wish to minimize fuel consumption, maximize maneuverability, etc.

Segment selection from a single portrait — a set of discretized trajectories — proceeds in two stages. Trajectories that do not touch the endpoint cells are quickly eliminated with simple tests. Full state-space versions of the remaining candidates are then reconstructed and tested using the optimality metric, narrowing the field to a single result. This operation

[6]To extend the highway route planning analogy: one chooses a length of interstate and then connects to it on smaller roads, backtracking if appropriate.

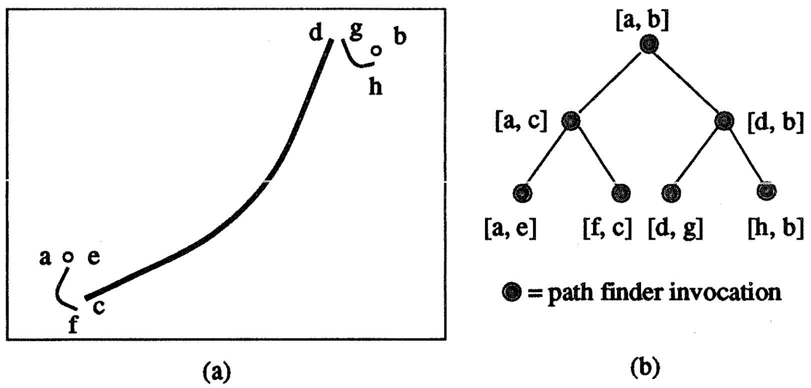

Figure 4: A segmented path (a) and the path finder calls that created it (b). [x, y] denote pairs of points to be connected at each pass; later calls are lower down in the tree and connect closer pairs than those above them

is repeated for each portrait in the stack, and the *core segment* of the ultimate reference trajectory is extracted from the overall winner.

The entire mapping/search process is then repeated inside the two endpoint cells, this time to connect the endpoints of the core segment to the control objectives. See figures 3 and 4. Note that this effects a finer-scale division of the endpoint cells[7] and exploration of the dynamics therein; this refinement can stem from dynamics as well as from control considerations. An example of the grid spacing that this dynamic revision can produce is shown in figure 5. The spacing reflects the locations of the objectives, as well as local differences in either trajectory spreading or sensitivity; the top right is either more turbulent or more responsive to the control parameter than the bottom left. Similar pictures arise — for similar reasons but via different patterns and reasoning — in [17] and in Multigrid[6].

The mapper/path-finder tandem continues to fill the gaps in the evolving partial path until the resulting ensemble of segments meets the specified tolerance.[8] *Phase Space Navigator*[25], which uses the output of the MAPS program described in the previous section, bypasses this type of layered approach by reasoning with flowpipes instead of individual trajectories. *Perfect Moment* converts the final collection of segments into a *recipe*, which contains a list of the parameter values and endpoints of each segment, plus the linearizations and sensitivity derivatives at the segment junctions. This recipe is passed to the on-line controller described in the next section.

If the search fails, it is repeated with a heuristically modified choice for the initial cell

[7]The cells are recentered as well; see the dashed rectangles in figure 3.

[8]Tolerance checking is not simply a matter of halting when the endpoint of the last segment found falls within a specified distance of the control destination. This (fairly involved) computation is described in section 5.4 of [3].

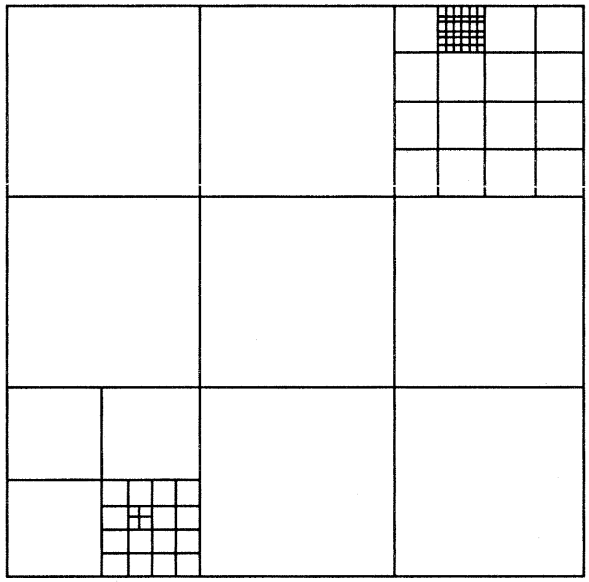

Figure 5: Variable-resolution grid: the local resolution is automatically adapted to the dynamics and control requirements in each region

size and the inter-step reduction. Should this still fail, the region is expanded (via the over-range parameter), together with the parameter range and the mapper's iteration depth. Of course, pathological systems exist wherein *no* set of mapping or path-finding parameters suffices to make the search succeed.

Perfect Moment reasons about large-scale dynamics in order to find large sections of the reference trajectory. Among other benefits, these tactics allow the program to find locally bad segments that lead to globally good paths.[9] This reasoning is implemented using the grid and the rules that manipulate it; this machinery not only tailors the scale of the exploration to fit the scale of the task, but also focuses the program's attention on the regions where it is most needed — because of interesting, complicated or useful dynamics or because of proximity to control objectives or to evolving partial paths.

On-Line Control

The on-line controller causes a system to follow a reference trajectory, performing the appropriate parameter value switch when the system state reaches each segment junction. Obviously, the timing and accuracy of these switches are critical. Since *Perfect Moment*'s ultimate goal is the control of real physical systems, it uses an additional, autonomous control device in an attempt to correct for such errors — a simple local-linear controller programmed with the linearization and sensitivity derivatives at each segment junction.

[9] e.g., driving east to an airport to catch a westward flight.

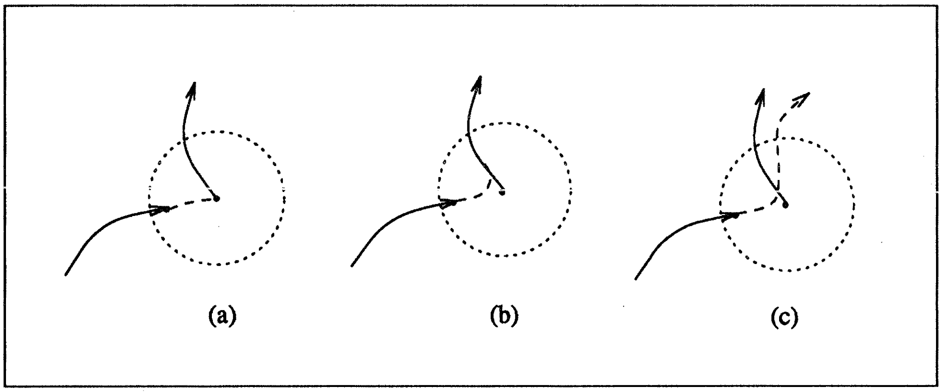

Figure 6: Control at the segment junctions: (a) ideal case (b) successful control (c) unsuccessful control. The dotted circle is the junction controller's range; the solid trajectories are the desired path segments and the dashed paths are the actual trajectories.

Patching together a collection of local *linear* controllers into a global control system has a long and rich tradition; it was pioneered by Kalman[14] in the 1950s and, recently, has even seen some AI applications[22]. The approach outlined here is different because it combines linear junction control with *nonlinear* global control and trajectory planning. The junction controller can also be used to stabilize the system at the objective, either directly or via OGY control of an unstable periodic orbit[19]. Figure 6 shows how such a controller works. The linearizations and sensitivities govern the size and shape of the controller's range, shown here (naively) as a circle. Upper bounds on achievable segment and path lengths depend intimately on the range at each junction, as well as on quantization error (via machine epsilon), Lyapunov exponents, integrator error, sensor and actuator accuracy, etc. See section 6.1 of [3] for more details. Model accuracy, the most pernicious of these problems, is discussed in more detail in [1].

Chaotic attractors are, oddly enough, robust with respect to non-trivial[10] amounts of *state* noise. Such a deviation bumps a point onto a nearby trajectory *where it would eventually end up at some point in the system's evolution anyway* — it does not change the character of the attractor, only the order in which the whorls are traced out. Together with the characteristic denseness with which a trajectory covers such an attractor, this leads to an interesting form of chaotic robustness. If the objective is on the attractor, *any* trajectory in the basin of that attractor will eventually pass within ϵ of it, for arbitrarily small ϵs. This property is used extensively in the phase-locked loop example in this paper; coupled with standard linear control and the properties of a chaotic attractor, it also forms the core of OGY control[19]. Note that the time required to approach a point on

[10] up to the size of the enclosing basin

the attractor is *effectively non-deterministic* — because of small, unavoidable errors and nonlinear amplification. A later development in OGY control, termed targeting[23], exploits nonlinear leverage to hasten this acquisition. The tactics and results in [23] are very similar to those of a two-pass run of the mapper/path-finder tandem (cf., the Lorenz example in figure 8). A major difference between *Perfect Moment* and OGY is that no recursive search for secondary switch points is performed in the latter — not surprising, as the algorithm has not been automated.

Examples

All of the examples in this section — the Lorenz system, the driven pendulum, and the phase-locked loop — are simulated. However, physical applications are the ultimate target of this work, so the models and controls in the second and third examples reflect the physical parameters of real mechanical and electronic systems that have been constructed as test cases for these techniques. Modeling and experimental error have caused some difficulties in the extension of the simulated results to actual physical systems[1].

The Lorenz System

State-space portraits generated from the Lorenz equations[15], which approximately describe convection in a sheet of fluid heated from below, show all the classic chaotic properties that give *Perfect Moment* its power. For low parameter values,[11] two stable fixed points exist; as the parameter is raised, these fixed points bifurcate into a chaotic attractor. These equations and two randomly chosen origin and destination points (**A** and **B** in figure 7) were presented to *Perfect Moment*.

The mapper constructed a set of state-space maps: initially at large parameter intervals in the range where the fixed points are stable and then at smaller intervals near the bifurcation point and in the ranges where any attractor (chaotic, limit cycle, or stable fixed point) approached the objective **B**. An initial grid size was then chosen and the collection of maps was searched for the shortest segment between the grid squares containing the origin and destination. The best choice turned out to be a section of a chaotic attractor; its Lyapunov exponent was then computed and used, along with other factors, to step down the cell size in the endpoint cells for the second search phase. The process was repeated only once more,

[11]The parameter is the coefficient r in equation (26) of [15]. We make no assertions about whether changing this parameter is either physical or practical; this purely mathematical example is presented mainly as a point of comparison and contrast to other published results. Lorenz himself explored the parameter space outside the range ($r \approx 1$) within which the equations are considered to be an accurate physical model.

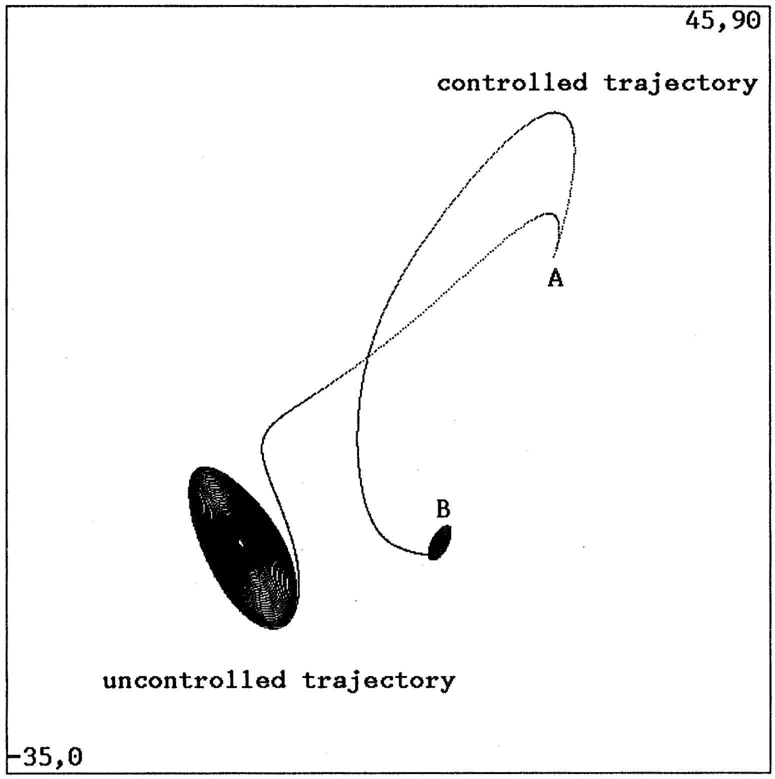

Figure 7: A reference trajectory to a fixed point in the Lorenz system

since a stable fixed point was found near the destination for another parameter value and the end of the first segment reached well into its basin of attraction. Both controlled and uncontrolled paths are shown in figure 7. Note the trajectory's initial move away from the control objective **B**. Since the origin is actually in the basin of attraction of the *other* fixed point — as indicated by the uncontrolled trajectory — the chaotic attractor segment may be viewed as a "bridge" over the basin boundary. This bridge also improves convergence by bypassing most of the tightly wound spiral around **B**.

Given a task with similar optimality criteria and different origin and destination points, the latter on a chaotic attractor, *Perfect Moment* constructed the reference trajectory shown in figure 8. This trajectory contains four segments: two chaotic attractor segments, and two (invisibly small) sections of trajectories that are relaxing to stable fixed points. As in the previous figure, the controller's first move — the short horizontal segment emanating

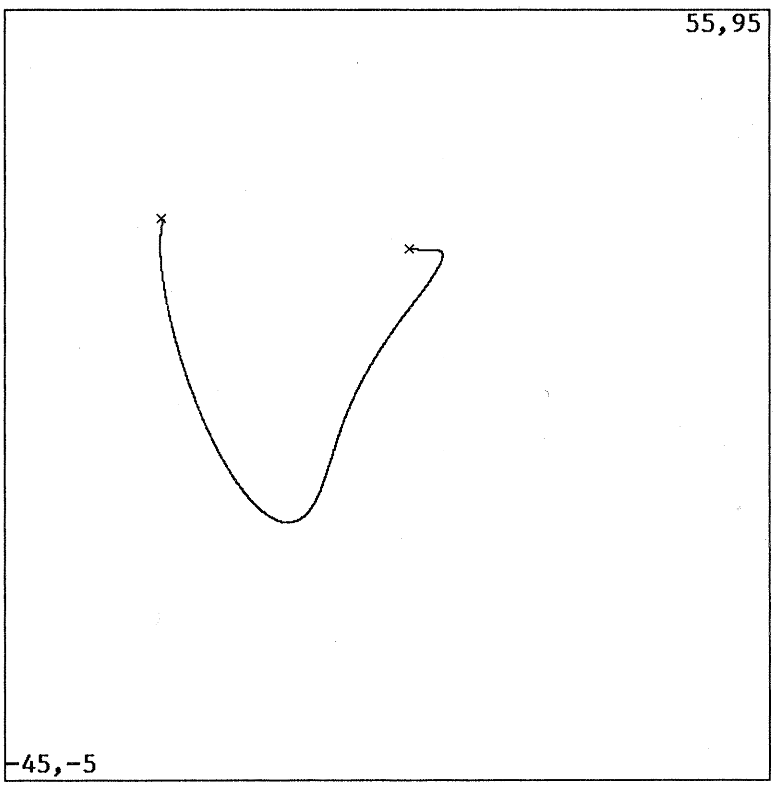

Figure 8: A reference trajectory to a point on a chaotic attractor in the Lorenz system

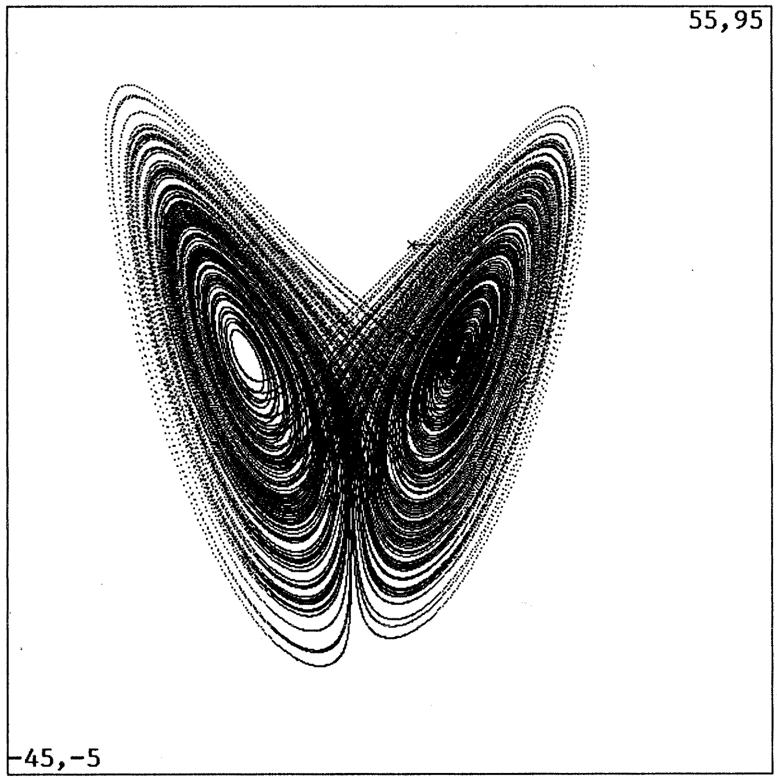

Figure 9: Time to acquire and achieve OGY control with the same initial conditions and requirements as the previous figure

from the right-hand cross — forces the system *directly away from* the control objective. This counterintuitive move is a calculated nudge that pushes the trajectory to the chaotic attractor segment that goes directly to the objective. This is essentially equivalent to the targeting of [23]. The contrast to the case *without* active target acquisition (figure 9, which uses the same conditions as figure 8 but follows the approach outlined in [19]) is striking: the acquisition time is improved by about a factor of 290, and the path is about 1/200th as long.

The Inverted Pendulum

The driven pendulum is arguably the most closely studied simple chaotic system (e.g., [7, 8, 12]); it has many practical applications, from robotics to offshore drilling platforms to earthquake-proofing of buildings. *Perfect Moment* was used to balance the pendulum

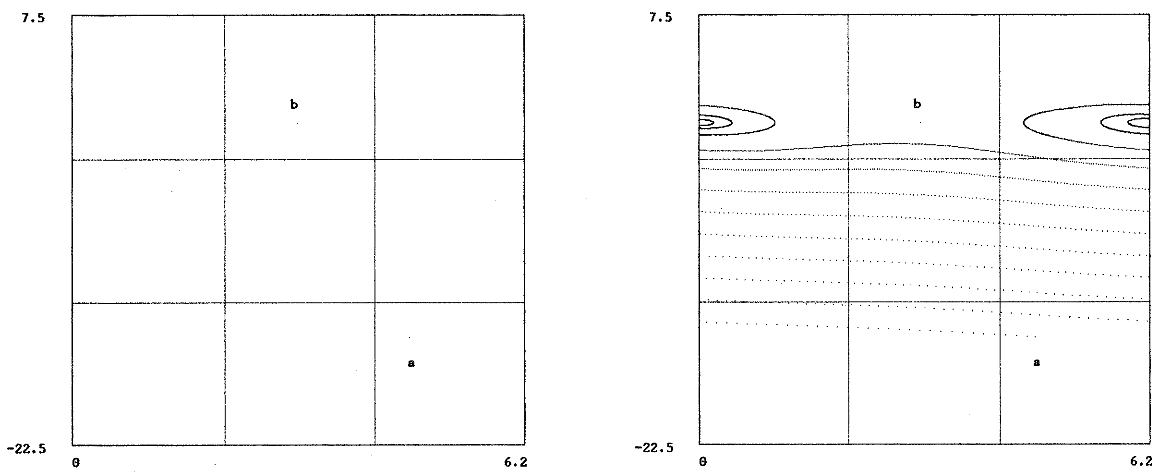

Figure 10: The setup for the task of balancing the driven pendulum at the inverted point: part (a) shows the origin and destination points and part (b) shows the uncontrolled trajectory from the origin. The vertical axis is the angular velocity ω and the horizontal axis is θ modulo 2π

inverted — at $(\theta,\ \omega) = (\pi,\ 0)$ — from some random initial condition $(-\pi/2, 15)$. These origin and destination points are labeled **a** and **b** on figure 10(a). If the pendulum is started from **a** with no applied torque, the uncontrolled system follows the trajectory shown in part (b) of the figure. The initially high kinetic energy ($\sim \omega^2$) dissipates over eight circuits of the cylinder, and the trajectory then oscillates down to the fixed point. At no time does it closely approach **b**.

The first path found by *Perfect Moment*, shown in part (a) of figure 11, contains two segments. The long segment, marked "p=15" in the figure, requires a very high driving torque ($\sim p^2$): 75% more than a *simple linear controller* would use to balance the pendulum at the inverted point, given the same starting conditions. If a measure of the torque is incorporated into the Scheme procedure that acts as the optimality metric, *Perfect Moment* demonstrates its ability to reason globally, constructing a reference trajectory (part (b) of figure 11) that "pumps" the state up from **A** using a much smaller torque over a much longer period. Allowed the same amount of torque, a linear controller would lift the pendulum up to about a 50 degree angle from the vertical, and then hold it there, unable to go higher.

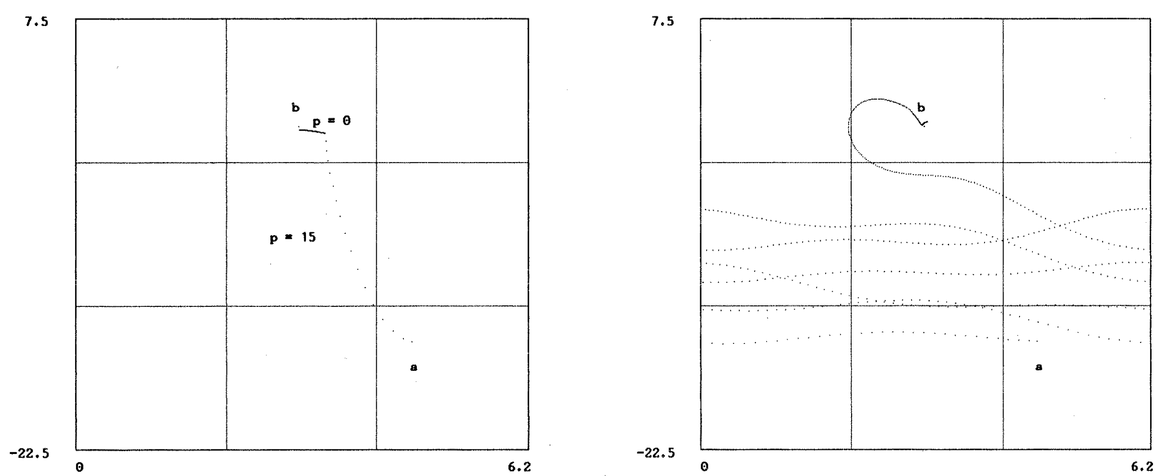

Figure 11: Reference trajectories that balance the driven pendulum: (a) fast trajectory with high-torque control (b) slower trajectory that uses less torque. The trajectory in part (b) "pumps" the state up from the initial condition over several cycles, attaining the control objective using one ninth the torque used in part (a) — but over 45 times slower. A linear controller, allowed this amount of torque, would not be able to balance the pendulum. Same axes as previous figure

The Phase-Locked Loop

A phase-locked loop (PLL) is an electronic circuit that tracks the frequency and phase of an input signal. A PLL can lock to an input sine wave whose frequency is within some *capture range* of its own internal oscillator's free-running frequency, and can remain locked to that signal over some (concentric) *lock range*. The latter is generally much wider than the former. In one particular class of these circuits, the differential equations that describe the evolution of the locking process are identical, within coefficient values, to those of the driven pendulum. The chaotic behavior of this circuit has been a topic of active research in the circuits community for some time[9]. Recently, synchronized chaos[20] has been induced in this system[10] to facilitate secure communications via transmission on a chaotic carrier[11].

Perfect Moment can be used to exploit the chaotic behavior in a different way: to improve the design of the circuit itself. Specifically, a suitable reference trajectory can broaden the capture range out to the lock range limit[4]. A second, variable-frequency drive is used to force the circuit's state from a starting point outside the capture range onto a chaotic attractor that overlaps the original lock range. When the trajectory enters that region, the additional drive is immediately turned off and the circuit is allowed to settle into lock according to its original unmodified dynamics. This usage is different in spirit from the two previous examples, as the control objective is not a single point but a wide range. However, the results — increased reachability due to a "strange attractor bridge" — are similar.

Conclusion

Chaotic systems are uniquely sensitive to small parameter and state pertubations, and yet exhibit characteristic structure (e.g., the geometry of their so-called *strange attractors*). This paper describes and illustrates a computer program that uses fast and accurate computation to synthesize paths through a chaotic system's state space that exploit these properties to accomplish otherwise-impossible control tasks. Nonlinear dynamics provides the mathematical tools used by these algorithms to choose values, tolerances, heuristics and limits for the selection of trajectory segments and their synthesis into useful reference trajectories. Many of the trajectories found by *Perfect Moment* are shorter and faster than those found by traditional control methods, make unreachable control objectives reachable, and improve convergence. The program builds these trajectories by reasoning about the dynamics at the scale dictated by the task, exploring counterintuitive moves, exploiting the denseness of chaotic attractors, and utilizing regions of *sensitive dependence on initial conditions* in a system's state space. It does not always outperform classical linear and nonlinear control techniques. In fact, it sometimes fails to find a path at all in a problem that is easily solved by the standard techniques. The converse is also true, however, making this type of technique a useful addition to the existing arsenal of control techniques.

Perfect Moment has a variety of shortcomings. Since it currently depends on presimulation of the system state, it cannot be applied to systems where the state variables are neither directly nor indirectly observable, nor can it adapt to bad models or react to time-varying systems. Gathering data from physical devices, rather than ODE models, would vastly reduce the effects of modeling problems; this investigation is currently underway in the author's group. An even-better solution would be to *plan* on a global scale and *track* on a local scale, extending the use of the linear controller from a patch around the switch point of figure 6 to a tube around the entire length of the path. The current incarnation of the program only handles systems that have a single control parameter; changing the code to relax this restriction would be easy, but the run time is exponential in the number of parameters. A few other important caveats are rigor and range of applicability: *Perfect Moment* uses heuristics extensively and chooses roughly optimum solutions by balancing several simultaneous tradeoffs and diminishing-return situations. It does not hold out for truly optimal solutions, but rather concocts a "good enough" one as quickly as possible. Thus, proving that a result is "optimal" — or even determining in advance whether or not it will succeed on a given problem — is virtually impossible.

The driving concept behind this approach to control of nonlinear and chaotic systems is to combine fast computers with deep knowledge of nonlinear dynamics to improve performance in a class of systems whose performance is rich but whose analysis is mathematically and computationally demanding.

Acknowledgements

The author is grateful to Harold Abelson, Brian LaMacchia, Gerald Jay Sussman, and the other members of the MIT Project on Mathematics and Computation for past and present support and encouragement.

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contracts N00014-85-K-0124, N00014-86-K-0180, and N00014-89-J-3202, an AAUW dissertation fellowship, and National Science Foundation grant MIP-9001651 and National Young Investigator Award CCR-9357740.

References

[1] E. Bradley. The driven pendulum: Theory, practice and implications for control. In

preparation.

[2] E. Bradley. A control algorithm for chaotic physical systems. In *First Experimental Chaos Conference*. World Scientific, 1991.

[3] E. Bradley. *Taming Chaotic Circuits*. PhD thesis, M.I.T., September 1992.

[4] E. Bradley. Using chaos to broaden the capture range of a phase-locked loop. *IEEE Transactions on Circuits and Systems*, November 1993.

[5] E. Bradley and F. Zhao. Phase space control system design. *IEEE Control Systems Magazine*, 13:39–46, April 1993.

[6] W. L. Briggs. *A Multigrid Tutorial*. SIAM Press, Lancaster, PA, 1987.

[7] P. J. Bryant and J. W. Miles. On a periodically forced, weakly damped pendulum. Part I: Applied torque. *Journal of the Australian Mathematical Society*, 32:1–22, 1990.

[8] D. D'Humieres, M. R. Beasley, B. Huberman, and A. Libchaber. Chaotic states and routes to chaos in the forced pendulum. *Physical Review A*, 26:3483–3496, 1982.

[9] T. Endo and L. O. Chua. Chaos from phase-locked loops. *IEEE Transactions on Circuits and Systems*, 35:987–1003, 1988.

[10] T. Endo and L. O. Chua. Synchronization of chaos in phase-locked loops. *IEEE Transactions on Circuits and Systems*, 38:1580–1588, 1991.

[11] J. Gullicksen, M. de Sousa Vieira, M. A. Lieberman, R. Sherman, A. J. Lichtenberg, J. Y. Huang, W. Wonchoba, M. Steinberg, and P. Khoury. Secure communications by synchronization to a chaotic signal. In *First Experimental Chaos Conference*. World Scientific, 1991.

[12] E. G. Gwinn and R. M. Westervelt. Fractal basin boundaries and intermittency in the driven damped pendulum. *Physical Review A*, 33:4143–4155, 1986.

[13] C. S. Hsu. *Cell-to-Cell Mapping*. Springer-Verlag, New York, 1987.

[14] R. E. Kalman. Phase-plane analysis of automatic control systems with nonlinear gain elements. *Transactions of the AIEE*, 73:383, 1955.

[15] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[16] D. Michie and R. A. Chambers. BOXES: An experiment in adaptive control. *Machine Intelligence 2*, 1968.

[17] A. W. Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of the 8th International Workshop on Machine Learning*, 1991.

[18] A. Newell, J. C. Shaw, and H. A. Simon. Preliminary description of General Problem-Solving-I (GPS-I). Technical Report Report CIP Working Paper 7, Carnegie Institute of Technology, Pittsburgh, PA, 1957.

[19] E. Ott, C. Grebogi, and J. A. Yorke. Controlling chaos. In *Chaos: Proceedings of a Soviet-American Conference.* American Institute of Physics, 1990.

[20] L. M. Pecora and T. L. Carroll. Synchronization in chaotic systems. *Physical Review Letters*, 64:821–824, 1990.

[21] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge U.K., 1988.

[22] E. P. Sacks. Automatic qualitative analysis of ordinary differential equations using piecewise-linear approximations. Technical Report AI-TR 1031, M.I.T. Artificial Intelligence Lab, March 1988.

[23] T. Shinbrot, E. Ott, C. Grebogi, and J. A. Yorke. Using chaos to direct trajectories to targets. *Physical Review Letters*, 65:3215, 1990.

[24] K. M.-K. Yip. KAM: Automatic planning and interpretation of numerical experiments using geometrical methods. Technical Report AI-TR 1163, M.I.T. Artificial Intelligence Lab, August 1989.

[25] F. Zhao. *Automatic Analysis and Synthesis of Controllers for Dynamical Systems Based on Phase-Space Knowledge.* PhD thesis, M.I.T., September 1992.